RAG System Development and Evaluation for Legal Domains

Supisara Kangval
3rd Year Data Science Student at University College London

Introduction

This report presents an end-to-end development and evaluation of a Retrieval-Augmented Generation (RAG) system designed to process and answer questions based on Thai legal documents. It covers the full pipeline from document preparation and chunking to embedding, retrieval, reranking, and prompting followed by performance evaluation using established metrics. Key challenges such as hallucination, irrelevance, and answer correctness are analysed using both quantitative scores and targeted visualisations. The report also includes threshold-based error categorisation and comparesons between different chunking methods. Overall, it provides a comprehensive assessment of the system's strengths, limitations, and directions for improvement.

Data Preparation

Data Collection

Legal documents were internally sourced, focusing primarily on three distinct legal topics:

- SD-01-01-R07: มาตรฐานการเก็บรวบรวมและรายงานข้อมูลอุบ-ติการณ์ด้านอาชีวอนามัย ความปลอดภัย (Codes for Collecting, Compiling, and Reporting Occupational Health and Safety Incident Data)
- SD-01-04-R01: เอกสารอ้างอิงคู่มือการปฏิบัติกฎพิทักษ์ชีวิตของเอสซีจี (Reference Documents: SCG Life Protection Practice Manual)
- SD-01-03-R01: มาตรฐานการทำงานบนที่สูง (Codes for Working at high Elevation Standard)

These documents serve as foundational data for the RAG system, ensuring accurate and contextually relevant responses tailored specifically to Thai legal domains.

Text Extraction

The extraction pipeline involved several carefully designed steps to retain text integrity and consistency:

Header and Footer Removal: each document underwent pre-processing to crop out unnecessary header bands and page footers, ensuring clean input for subsequent analysis.

Document Segmentation: by leveraging each document's table of contents, the texts were systematically split into separate files corresponding to individual topics, reducing context loss when splitting them into chunks.

LLM-driven Content Extraction: a specially crafted prompt was used to instruct the Large Language Model (LLM) to extract text verbatim, strictly preserving the original languages (primarily Thai with occasional English terms). This ensured the legal specificity and contextual nuance remained intact.

Verification of Extracted Data

Extracted texts underwent rigorous quality assurance, involving engineers who specifically checked for:

Accuracy: Confirming the extracted content faithfully matched the original documents.

Completeness: Ensuring no critical legal clauses or paragraphs were omitted during extraction.

Handling of Figures

Figures within legal documents pose unique extraction challenges due to varying complexity:

LLM-based Figure Interpretation: Simple and moderately complex figures were explained by prompting an LLM to summarize their contents and relevance succinctly.

Irrelevant Figure Elimination: non-essential figures or those not contributing directly to the legal context were identified by LLM analysis and subsequently removed from the dataset.

Complex Figures: certain diagrams and figures exceeded the interpretative capabilities of the LLM and required specialized legal domain expertise. These figures are currently marked and temporarily excluded from the dataset, pending domain-expert review.

Challenges with Thai Language Data

Thai language presents specific challenges to RAG systems, particularly regarding tokenization and spelling accuracy. Given the continuous nature of written Thai (lacking explicit spaces between words), a dedicated tokenizer was necessary. This tokenizer helps ensure the model accurately interprets and retrieves text segments effectively, thus reducing search ambiguity.

The tokenizer selected is pythainlp.word_tokenize with the newmm engine which is the default engine that utilizes a dictionary-based approach combined with the Maximum Matching algorithm, which accurately segments Thai text into words despite the absence of spaces [1]. This tokenizer is used in the length function to measure chunk_size in words rather than characters. Separators are applied in order from paragraph breaks, line breaks, spaces, punctuation, and zero-width or fullwidth symbols. This ensures splits occur at natural boundaries before falling back to character-level splitting when necessary.

Embedding into Vector Database with Pinecone

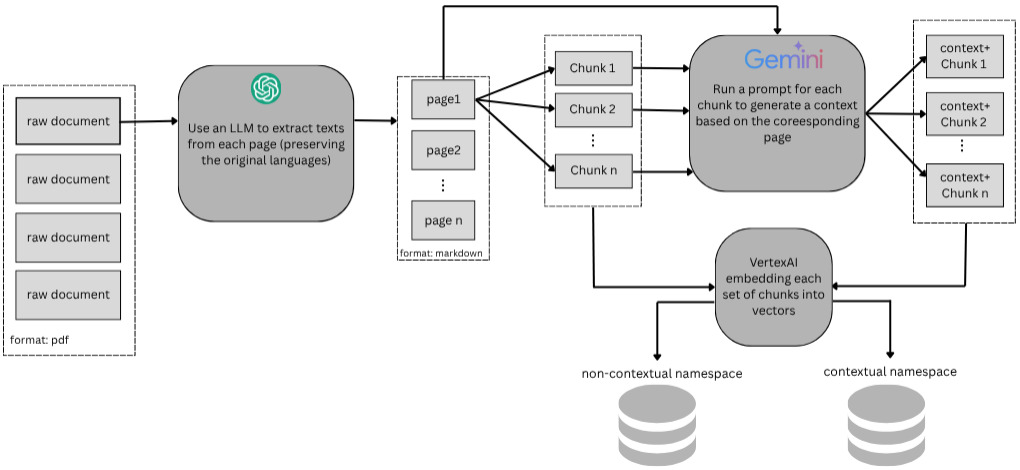

Figure 1 shows the workflow of data preparation process

Pinecone provides robust functionality essential for building and managing vector databases. It supports integration with multiple embedding models, allowing flexibility in converting textual data into numerical vector representations suitable for semantic search.

Index and Namespace

In Pinecone, data is organized within an **index**, which serves as a container capable of storing multiple separate namespaces. Name-spaces act as sub-databases within a single index, allowing efficient management and isolation of datasets for different applications or document groups. For this project, the index is structured to contain distinct namespaces for non-contextual and contextual chunks, allowing for a comparative evaluation of RAG performance across the two chunking strategies.

Chunk Size and Overlap Choices

The chunking strategy for embedding was configured with a chunk_size of 500 tokens and an overlap of 150 tokens.

We will explore, in the later section, contextual chunking where we employ an LLM to generate context for each chunk based on the part of the document it comes from as an alternative.

Data Retrieval

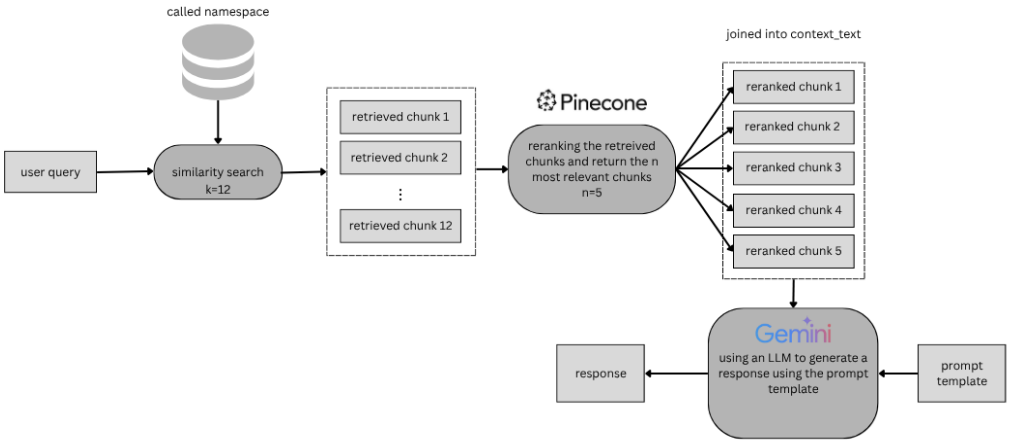

Figure 2 shows how data is retrieved and generated.

Process of Retrieval

Figure 2 illustrates a RAG pipeline optimized for legal document queries. When a user submits a query, a similarity search retrieves the top 12 relevant chunks from a vector database. These are then re-ranked using Pinecone to identify the 5 most relevant chunks. Only chunks with a relevance score above the chosen threshold are retained to reduce unnecessary context. The selected chunks are concatenated into a context text and passed into a prompt template. Gemini, a large language model, then uses this structured prompt to generate a context-grounded response, enhancing both accuracy and efficiency by filtering irrelevant or misleading information. The variables are defined as follows:

k: the top *k* relevant chunks from the vector database using a similarity search.

n: is the top *n* relevant chunks from the k retrieved chunks using a reranker.

Threshold: a value between 0 and 1; only chunks with a reranker relevance score greater than or equal to this threshold are included in the context passed to the LLM for answer generation.

Reranking

Standard vector search methods inherently possess limitations – loss of information resulting from compression into single vector embeddings. Consequently, relevant document chunks might fall below the initial retrieval cut-off defined by the *top_k* parameter. In this section, *top_k* refers specifically to the number of retrieved contexts passed to the LLM for generation and should not be confused with the *k* value used in earlier parameter tuning experiments. One straightforward solution would be increasing *top_k*; however, this is constrained by the LLM's context window limits. An excessively large retrieval set negatively impacts the LLM's recall performance which is the ability of the LLM to effectively retrieve information diminishes as document positioning within the prompt increases.

To address this challenge, a reranker (cross-encoder model) is introduced, forming a two-stage retrieval pipeline. In the first stage, cosine similarity search (bi-encoder model) identifies a broader candidate set from the vector database. In the second stage, the reranker carefully evaluates and reorders these retrieved chunks based on their actual relevance to the query and outputs the top *n* most relevant chunks.

Reranking significantly outperforms traditional embedding-based retrieval alone because it compares the user query directly against each retrieved chunk pairwise, rather than relying solely on precomputed embedding similarity. By implementing a cross-encoder model, the reranker captures deeper semantic relationships between the query and each document chunk, resulting in more precise final rankings and higher-quality retrieval outcomes [2].

Choice of *k* and *n* Values

In the later section, we will explore how varying retrieval parameters *k* (the number of top chunks retrieved via similarity search) and *n* (the number of top chunks retained after reranking)

affects overall system performance. Tuning these values is important because they directly influence the balance between retrieving enough relevant context and avoiding unnecessary or distracting information. By systematically evaluating different combinations, we aim to understand their impact on different metrics.

Generation

Principle of Prompt Engineering

Prompt engineering for legal document-based RAG system focuses on clarity, constraint, and context relevance. Given the complexity and precision required in legal language, prompts must guide the LLM to generate answers strictly grounded in retrieved content. Techniques like Chain-of-Thought (CoT) are used to break down legal reasoning into logical steps, improving answer coherence. Additionally, incorporating relevance scores into the prompt helps the LLM prioritise high-quality context. The goal is to maximize factual accuracy and reduce ambiguity in responses within tightly scoped legal domains.

Evaluation

Explanation of Variables Used in Evaluation Calculations

The RAGAs evaluation framework requires four key variables to accurately measure RAG system performance:

user_input: The original question or query submitted by the user.

retrieved_context: Contextual chunks retrieved from the knowledge base by the retriever and reranker, used to generate the response.

reference (ground_truth): The ideal answer assumed to be correct and complete, serving as a benchmark for accuracy. Note that in this report, the terms reference and ground_truth (GT) will be used interchangeably.

response: The final output or answer generated by the RAG system.

To create the evaluation test set, we employed an LLM to generate three questions and corresponding ground-truth answers for each cleaned markdown (md) document, assuming these answers as complete and accurate following with verification by an engineer, though ideally this process should be conducted by a domain expert. At the integration layer, we compiled lists of user_input (questions) paired with their corresponding GT entries, which in this case were identified as threats to validity. For each question, we ran the retrieval and generation phases to produce both retrieved_context and response, storing these results in dictionaries for subsequent evaluation. Each entry in the test set is evaluated independently as a single-turn sample, meaning that each row is assessed as a standalone response without incorporating any prior conversational history. It is also important to note that, for certain evaluation metrics, elements such as responses and ground-truth answers are decomposed into smaller units of text referred to as 'claims'.

Definitions and Computational Approach

Evaluation using RAGAs assesses two main aspects of the RAG system: Retrieval Quality and Generation Quality. The scores are values between 0 and 1, indicating the degree to which each metric is reflected in the results [3].

Evaluation of RAG Retrieval:

Context Precision: Measures the proportion of retrieved context chunks that are relevant to answering the user's query over all the n retrieved chunks. Higher precision indicates fewer irrelevant retrieved chunks.

$$context\ precision\ = \frac{\Sigma(Presision@k)}{number\ of\ relevant\ items}$$

$$Precision@k\ = \frac{True\ Positive@k}{True\ Positive\ @k + False\ Positive@k}$$

Context Recall: Measures how well the retrieval captures all relevant context necessary to address the query by breaking GT into a set of claims and each claim is compared against retrieved context whether it exists or not. Higher recall means fewer relevant chunks were missed.

$$context\ recall\ = \frac{GT\ claims\ that\ can\ be\ attributed\ to\ context}{number\ of\ claims}$$

Evaluation of RAG Generation:

Faithfulness: Assesses how accurately the generated response adheres to information present in the retrieved context. To measure this, each response is broken down into claims and compared against retrieved context. Higher faithfulness

indicates minimal generation of unsupported claims, in other word, minimal hallucination.

$$faithfulness = \frac{|number\ of\ claims\ that\ can\ be\ inferred\ from\ the\ given\ context|}{number\ of\ claims\ in\ the\ answers}$$

Answer Relevancy: Evaluates how directly the generated answer addresses the user's original query, regardless of correctness. Starting from generating N questions based on the response then cosine the vector similarity of generated questions with the original questions. Higher relevancy indicates the answer directly corresponds to the query intent.

$$answer\ relevancy = \frac{\Sigma[cos(Embedding_{G_i}, Embedding_{o_i})]}{N}$$

Answer Correctness: Measures the factual accuracy together with semantic similarity of the response compared to the GT answer. By default, the evaluation score is weighted such that 75% is derived from factual correctness, measured primarily using the F1 score, and 25% from semantic similarity [4].

Factual Correctness: Evaluates the factual accuracy of the generated response with the GT. There are three modes of factual correctness:

1. Precision: proportion of True Positive (TP) claims over TP claims and False Positive (FP) claims.

$$Precision = \frac{TP}{(TP + FP)}$$

2. Recall: proportion of TP claims over TP and False Negative (FN) claims.

$$Recall = \frac{TP}{(TP + FN)}$$

3. F1 score: measurement that contributes both precision and recall.

$$F1\ score = \frac{2\ x\ Precision\ x\ Recall}{(Precision + Recall)}$$

$$= \frac{TP}{(TP + 0.5(FP + FN))}$$

There are also attributes like atomicity that control how much a sentence is broken down and coverage controls how comprehensively the response [5].

Semantic Similarity: Vectorises the GT and response using the same embedding model then computes the cosine similarity between them [6].

Practical Considerations and Evaluation Refinement

Test set preparation

We encountered a parsing issue when reading the CSV file, as commas within the user_input and reference fields were mistakenly treated as delimiters. This caused misalignment of columns and incorrect row formatting. To resolve this, we changed the CSV separator to a semicolon (;), which is less likely to appear in the text and preserves the structure during processing.

Limitation in Quality of GT Impacting Answer Correctness

A significant challenge encountered was the absence of domain expert verification for generated ground-truth data and the quality of test questions. This limitation negatively impacted the Answer Correctness metric, particularly due to the influence of the factual_correctness score, which heavily penalises FP and FN. Given that the factual correctness component accounts for approximately 75% of the overall answer correctness metric, scores were disproportionately lowered. To address this, we adjusted metric weighting, placing greater emphasis on semantic similarity and reducing the impact of the factual correctness score–75% semantic similarity and 25% factual correctness. This provided a more balanced and fair evaluation of answer correctness in the absence of verified expert data.

documents	scores			
	answer_correctness	answer_correctness_reweight	factual_correctness	semantic similarity
SD-01-01-R07	0.6222	0.7940	0.5363	0.8799
SD-01-03-R01	0.7406	0.8597	0.6810	0.9193
SD-01-04-R01	0.6558	0.8391	0.5641	0.9308
Total Average	0.6728	0.8309	0.5938	0.9100

Figure 3 presents the evaluation results for answer correctness before and after metric reweighting, along with the individual factual correctness and semantic similarity scores that contribute to the overall answer correctness metric.

The table above shows that reweighting the answer_correctness metric increases the overall average score from 0.6728 to 0.8309, a gain of 23%. The factual correctness average (0.5938) indicates moderate grounding, while high semantic similarity (0.9100) shows strong alignment with *reference* meaning. This adjustment reduced the disproportionate penalty from factual correctness when expert-verified references were unavailable for a fairer assessment. However, it places greater weight on meaning alignment, so factual inaccuracies have less influence on the overall score compared to the original weighting.

Factual Correctness Refinement

To make the assessment fairer, we adjusted the evaluation approach. We used high atomicity, so each legal point is judged on its own rather than marking an entire answer wrong for one missing detail. We also applied high coverage, allowing partially correct answers to score well if they include key legal ideas, even if they miss smaller points.

Finally, by looking at precision and recall, we could see whether errors came from adding wrong information (FP) or missing important parts (FN). These adjustments help us judge the system more fairly, given the imperfect reference.

Evaluation Results

Comparison between results from non-contextual and contextual chunking

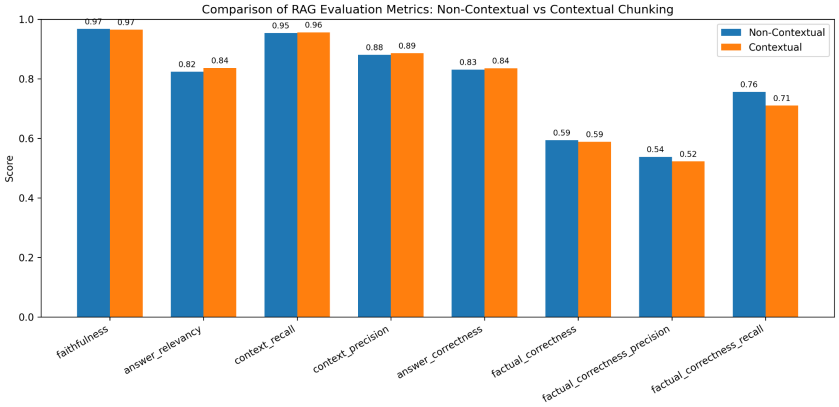

Figure 3 illustrates comparisons of evaluation results across the three documents between non-contextual and contextual chunking methods

The figure above presents a comparative analysis of RAG performance between non-contextual and contextual chunking methods. The results indicate that both approaches yield nearly identical scores across evaluation metrics. As previously noted, the marginal performance gain offered by contextual chunking does not justify the additional computational cost it incurs. Given this observation, we proceed with the non-contextual (standard) chunking method for the remainder of the evaluation, as it offers comparable performance with significantly lower computational overhead.

Impact of *k* and *n* on Retrieval Performance

k	n	# faithfulness	# answer_relevancy	# context_recall	# context_precision	# answer_correctness	# average
12	3	0.9674	0.8238	0.9539	0.8805	0.8309	0.8913
12	5	0.9678	0.8380	0.9405	0.8893	0.8314	0.8934
12	8	0.9498	0.8359	0.9444	0.8801	0.8253	0.8871
24	3	0.9632	0.8446	0.9295	0.9061	0.8320	0.8951
24	5	0.9587	0.8328	0.9646	0.8965	0.8313	0.8968
24	8	0.9672	0.8387	0.9639	0.8615	0.8352	0.8933

Figure 4 shows metric scores across *k* and *n* retrieval configurations.

The table above illustrates the average scores across all three documents under varying values of *k* and *n*. Overall, there is no significant difference in performance between configurations, as the average values across all evaluated metrics remain relatively consistent.

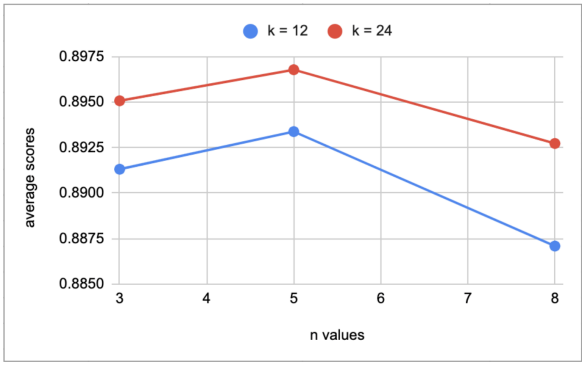

Figure 5 shows average scores comparison by *k* and *n* configurations.

A closer examination reveals that increasing *n* from 3 to 5 shows slight improvements in performance. However, when *n* reaches 8, average scores decline for both *k* = 12 and *k* = 24. The score trends across varying *n* values are broadly consistent between the two *k* settings. This analysis indicates that the optimal configuration is achieved with *n* = 5 and *k* = 24, yielding the highest overall performance across the evaluation metrics.

In a practical setting, however, we selected *k* = 12, *n* = 5. The justification being that it delivers a strong balance between performance and efficiency. Lower *k* values reduce computational cost since fewer neighbours in the vector database are considered. While higher *k* values improve robustness, it increases retrieval latency, memory usage, and the chance of including irrelevant context [7]. Here, *k* = *12* achieves robust performance comparable to *k* = *24* while significantly lowering processing cost.

Results and Interpretation

	SD-01-01-R07	SD-01-03-R01	SD-01-04-R01	Total Average
context_precision	0.83	0.93	0.89	0.88
context_recall	0.92	0.98	0.96	0.95
faithfulness	0.98	0.93	0.98	0.97
answer_relevancy	0.79	0.82	0.86	0.82
answer_correctness	0.79	0.86	0.84	0.83
overall_score	0.86	0.91	0.91	0.89

Figure 6 shows the evaluation metrics per document and overall Average

The RAG system demonstrates strong overall performance, with an average score of 0.891, indicating that the architecture is both robust and reliable in retrieving and generating contextually relevant, factually grounded answers from legal texts.

Faithfulness (0.967) is notably high across all documents, indicating that the generated responses remain closely tied to the retrieved context. This affirms the effectiveness of our two-stage retrieval approach and the use of a score threshold (≥ 0.3), which likely minimises hallucinated content.

Context Recall (0.954) is also exceptionally strong. This suggests that the retriever consistently captures most of the necessary information required to

answer the user queries. The selection of k=12 followed by reranking appears to strike a good balance between breadth and depth of retrieval.

Context Precision (0.880) indicates that many retrieved chunks are relevant and contribute meaningfully to the response, with minimal noise.

Answer Correctness (0.831), while still relatively high, there are areas for improvement. The slight drop in correctness can be attributed to limitations in the quality of the LLM-generated GT, which was not reviewed by domain experts. After the practical adjustment justified earlier, the answer_correctness shows a satisfying result.

Answer Relevancy (0.824), though strong, reveals a potential opportunity to improve how the LLM interprets user intent. Refinement in prompt chaining, better context structuring, or further prompt engineering could enhance alignment.

Overall, the evaluation confirms that our RAG system performs reliably across multiple legal documents, with consistent scores in both retrieval and generation metrics.

Examination of Impact of Factual Correctness

document	Factual Correctness Scores		
	F1	Precision	Recall
SD-01-01-R07	0.5363	0.492	0.6387
SD-01-03-R01	0.681	0.6314	0.8343
SD-01-04-R01	0.5641	0.49	0.7948
Total Average	0.5938	0.5378	0.7559

Figure 7 shows a table of factual correctness evaluation metrics per document.

The factual correctness evaluation reveals a moderate performance across the three documents. The average F1 score is 0.5938, indicating that while the system balances precision and recall reasonably well, there is still room for improvement. The average precision is 0.5378, showing that a notable portion of the generated claims are FP claims. Meanwhile, the average recall is 0.7559, suggesting the model captures a less portion of TP claims but may over-generate and include inaccurate information. The relatively low precision compared to recall reflects a tendency to prioritize coverage over accuracy, which could reduce reliability in high-stakes applications.

As discussed earlier, the limitation of the data and test set quality directly affect the significant low factual correctness results. We examine the potential correlations of factual correctness degradation and other metrics with certain chosen thresholds to determine the aspects of flaws of our RAG.

Analysis of Errors Patterns and Their Relationships

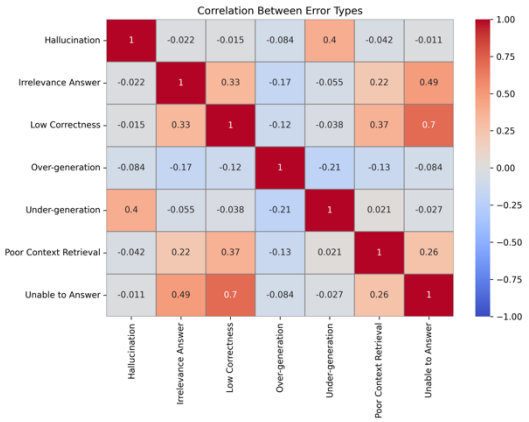

Figure 8 shows heatmap of correlation between error types.

This investigation explores the interrelationships between different error types in RAG-generated responses using a correlation heatmap.

The thresholds were selected to identify key weaknesses in the generated responses.

Hallucination is flagged when faithfulness < 0.6, indicating the presence of content not grounded in the retrieved context.

Irrelevance is identified when answer_relevancy < 0.5, suggesting the response does not adequately address the user's query.

Low Correctness is defined as answer_correctness < 0.6, reflecting insufficient alignment with the expected answer.

FP Margin refers to the degree to which factual_correctness (recall) exceeds factual_correctness (precision). A margin greater than 0.3 is used to flag potential over-generation and potential under-generation otherwise.

Over-generation (recall > precision) occurs when the model outputs more information than warranted, including excessive, irrelevant, or unsupported details beyond what mentioned in the GT.

Under-generation (precision > recall) is when the model omits necessary information, producing an incomplete or insufficient answer that fails to cover all relevant points supported by the context.

Poor Context Retrieval is identified when either context_recall or context_precision falls below 0.5,

indicating that the retrieved context is of insufficient quality.

Unable to Answer refers to cases where the model explicitly states that it cannot provide an answer based on the given information and responses with exactly *"ไม่สามารถตอบได้จากข้อมูลที่ให้มา"*.

The heatmap reveals that the correlation between different error types in the evaluation dataset. “Unable to Answer” has a strong positive correlation (0.70) with “Low Correctness,” indicating that most unanswerable responses are also judged to have low correctness. There is also a moderate correlation (0.49) between “Unable to Answer” and “Irrelevance,” suggesting overlap between unanswerable outputs and irrelevant answers. “Hallucination” is largely uncorrelated with other error types, except for a moderate correlation (0.40) with “Under-generation,” implying that missing information may sometimes be paired with hallucinated content. It can be noticed that there is a weak positive correlation between “Unable to Answer” and **“**Poor Context Retrieval*”* (*0.26*), suggesting that many “Unable to Answer” cases are not primarily due to missing or noisy retrieved content.

Conclusion

This report successfully presents the development and evaluation a Retrieval-Augmented Generation (RAG) system tailored for SCG Thai legal documents, covering the entire pipeline from pre-processing to final output evaluation. The system demonstrates strong performance across key metrics particularly in faithfulness, context recall, and context precision, proving effectiveness in minimizing hallucinations and indicating high reliability in retrieving and generating grounded answers.

Despite these strengths, the *answer correctness* and *relevancy* scores highlight areas for improvement. A major limitation was the lack of expert-verified questions and GTs, which negatively impacted factual correctness. Heatmap analysis further revealed some interesting insights that hallucination correlates with under-generation, and irrelevance often aligns with incorrect answers suggesting that retrieval strategy and prompt interpretation could be improved. It is also important to highlight the weak correlation between unable to answer and poor context retrieval.

Future work could involve conducting an additional statistical test, such as ANOVA to confirm whether performance differences between retrieval configurations are statistically significant. Moreover, an analysis could be done to identify patterns in questions that consistently yield lower scores or remain unanswered. This may reveal gaps in the retrieval corpus, reranking effectiveness, or LLM reasoning, together with implementing an LLM can help with guiding targeted improvements in data coverage, retrieval filtering, and prompt design to address specific weaknesses and improve overall system robustness.

Bibliography

[1] Anon, “pythainlp.tokenize — PyThaiNLP 2.0.3 documentation,” Pythainlp.org,, 2017. [Online]. Available: https://pythainlp.org/docs/2.0/api/tokenize.html.

[2] “Rerankers and Two-Stage Retrieval | Pinecone,” www.pinecone.io, [Online]. Available: https://www.pinecone.io/learn/series/rag/rerankers/.

[3] L. Monigatti, “Evaluating RAG Applications with RAGAs - TDS Archive - Medium,” Medium, 2023. [Online]. Available: https://medium.com/data-science/evaluating-rag-applications-with-ragas-81d67b0ee31a.

[4] Anon, “Answer Correctness | Ragas,” Ragas.io, 2023. [Online]. Available: https://docs.ragas.io/en/v0.1.21/concepts/metrics/answer_correctness.html.

[5] Anon, “Factual Correctness - Ragas,” Ragas.io, 2025. [Online]. Available: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/factual_correctness/.

[6] M. Ozkaya, “Understanding Semantic Meaning and Similarity Search: Cosine Similarity and Euclidean Distance,” 3 December 2024. [Online]. Available: https://mehmetozkaya.medium.com/understanding-semantic-meaning-and-similarity-search-cosine-similarity-and-euclidean-distance-666fcb5911ea. [Accessed 13 August 2025].

[7] D. Patel, “Balancing Computational Cost vs. Selecting the Best K Value in K-Nearest Neighbors (KNN),” Medium, 2024. [Online]. Available: https://medium.com/@helloitsdaksh007/balancing-computational-cost-vs-selecting-the-best-k-value-in-k-nearest-neighbors-knn-0ef9486a8404 .

Appendix A: Prompt Template

Prompt template for context generation:

```
prompt = ChatPromptTemplate.from_template("""
You are a domain expert in thai law for a cement company.
Your task is to **write a short, precise context summary in thai**
of a document chunk to improve its retrieval relevance.

    STRICT INSTRUCTIONS:
    - the summary needs to be in thai
    - Do **not** use phrases like "This chunk discusses",
    "This section is about", or similar.
    - Your response **must only contain the summary**,
    with **no preambles or extra explanation**.
    - Focus on **factual content**: key materials, processes,
    formulas, or principles.

    ---

    Reference Document:
    <document>
    {document}
    </document>

    Target Chunk to Summarize:
    <chunk>
    {chunk}
    </chunk>

    ---

    Write **2-3 sentences** that:
    1. Capture the main topic of the chunk.
    2. Include any technical terms, process steps, or
    formula names if available.
    3. Seamlessly integrate the chunk into the broader
    document's topic.

    Your answer should sound like a snippet from a well-written
    technical summary—not like a meta description.

    Output only the summary text.
    Context:
    """)
```

Prompt template for response generation:

```
law_prompt = """
# ROLE:
You are a legal expert specializing in Thai cement
industry regulations.

# TASK:
Your task is to answer the user's question **using only
the legal context provided below**, which is extracted
from official Thai legal documents. Each block of context
is accompanied by a relevance score from 0 to 1.

- Use information from **higher-scoring blocks first**.
- If the information is insufficient to provide a complete
answer, state: "ไม่สามารถตอบได้จากข้อมูลที่ให้มา" ("Cannot answer
based on the provided information.")

# LEGAL CONTEXT:
<context_blocks>
{context}
</context_blocks>

# USER QUESTION:
<question>
{question}
</question>

# RESPONSE INSTRUCTIONS:
- Provide a complete and legally accurate answer based
only on the content above.
- Do not infer, assume, or speculate beyond the provided text.
- Do not reference the existence of "the document" or any source.
- Use **formal, precise Thai legal language** suitable for
expert legal communication.
- Focus strictly on **legal relevance and clarity**. Avoid
rhetorical filler or generalizations.

# INTERNAL REASONING (Do not include this in the final answer):
Think step-by-step which context blocks are most relevant to the
user's question and extract only the directly applicable legal points.


# OUTPUT:
Respond with a legally sound answer grounded in the relevant legal
context above. If a full answer is not possible, respond with:
"ไม่สามารถตอบได้จากข้อมูลที่ให้มา"
"""
```

Appendix B: Reproducibility

To ensure transparency and enable replication of this evaluation, we document all key experimental parameters and settings below.

1. Model and Embedding Versions

- LLM model: gemini-1.5-flash
- LLM model provider: google_vertexai
- Temperature = 0.1
- Embedding model: models/gemini-embedding-001
- Reranker model: bge-reranker-v-m3

2. Retrieval and Context Configuration

- k = 5
- n = 12
- threshold (post-filter reranked score) = 0.3

3. Environment and Dependencies

The experiment relies on the following core Python libraries and frameworks:

- langchain, langchain_community, and langchain_core for prompt orchestration and vector store integration
- langchain_google_genai for Google Generative AI embeddings
- pinecone for vector database backend
- ragas for automated RAG evaluation metrics
- datasets for managing inputs and ground truth
- pythainlp for Thai language tokenization

Document chunking:

- Chunk size: 500 characters (based on RecursiveCharacterTextSplitter)
- Overlap: 150

Top-k contexts are retrieved using variable k values, where:

- k = 12 or k = 24 refer to how many document chunks are retrieved before reranking.
- Contexts are then reranked by a reranker and the top n reranked chunks are selected (n = 3, 5, or 8) for final LLM input.

4. Metric Evaluation

The evaluation uses the ragas.evaluate() function with the following metrics:

- answer_correctness
- faithfulness
- answer_relevancy
- context_precision
- context_recall
- factual_correctness_f1
- factual_correctness_precision
- factual_correctness_recall

The evaluation results are saved as a CSV or DataFrame for analysis.